
Pengenalan Sains Komputasi - MonteCarlo

Ridlo W. Wibowo || 20912009

November 5, 2012

Problem.
Buatlah program untuk mengenerate bilangan random menggunakan algoritma *Linear Congruential Generator* (LCG), lalu gunakan untuk menghitung integral tentu dari fungsi berikut ini:

$$f(x) = \sqrt{1 - x^4}$$
$$f(x) = \sin(\sqrt{x})$$
$$f(x) = e^{x^2}$$

Documentation.
Linear Congruential Generator
Generator ini merupakan contoh paling sederhana dari generator penghasil bilangan random. Perumusan yang digunakan adalah:

$$r_{n+1} = (a \times r_n + c) \mod m,$$

dengan
$r_0 =$ seed
$r_1, r_2, r_3, ... =$ random numbers
a, c, m are constants (multiplier, increment, and modulo)
Jika kita dapat memilih konstanta a, c, dan m di atas dengan teliti, maka kita akan mendapatkan generator bilangan random uniform antara 0 hingga $m - 1$.

Kekurangan dari LCG adalah r_n dan r_{n+1} tidak bebas sebagai bilangan random. Kita dapat memprediksi r_{n+1} apabila r_n diketahui, hal ini menyebabkan LCG tidak aman secara *cryptography*. Namun LCG cukup bagus untuk pekerjaan selain berhubungan dengan keamanan, misalnya integrasi montecarlo, selain itu algoritmanya sangat mudah diterapkan dalam bahasa pemrograman.

Pemilihan konstanta untuk LCG dapat kita tentukan sendiri secara hati-hati, namun kita dapat mencoba menggunakan konstanta yang sudah biasa digunakan sebagai *library* (walaupun ada library yang buruk, misalnya dari BSD, yang hingga akhirnya freeBSD mengubahnya, namun openBSD dan netBSD masih tetap menggunakannya). Beberapa telah dicoba, misalnya formula dari Microsoft berikut:

$$\begin{aligned} state_{n+1} &= 214013 \times state_n + 2531011 (\mod 2^{31}) \\ rand_n &= state_n \div 2^{16} \end{aligned}$$

$rand_n$ berada pada rentang $0 - 32767$. Walaupun rentangnya kecil (resolusinya rendah) namun pengulangan sequencenya cukup besar, yakni akan terjadi setelah $2^{31} - 1$ (memungkinkan keluar angka yang sama namun sequencenya berbeda). Kemudian untuk membuat *style* penggunaan bilangan random mirip yang digunakan pada bahasa C/C++ maka dibuat program header (`lcg.h`) sebagai berikut:

```
/*******************************************/
/* linear congruential generator           */
/* copyleft (c). Ridlo W. Wibowo           */
/* based on Microsoft formula              */
/*******************************************/
#ifndef LCG_H
#define LCG_H
#include <math.h>
#define MAX_RAND 32767

/* default seed */
static unsigned long int state = 1;

/* change seed */
void rseed(unsigned int s){
    state = s;
}

/* Linear Congruential Generator */
int randu(){
    unsigned int multiplier = 214013;
    unsigned int increment = 2531011;
    unsigned int modulo = pow(2,31);

    state = (state*multiplier + increment)%modulo;
    return state/pow(2,16);
}
#endif
```

Tes penggunaan (dibandingkan dengan fungsi `rand()` dari `stdlib.h`)

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
#include "lcg.h"
using namespace std;

int main(){
    //srand(time(NULL));
    for (int i=0;i<5;i++){
        cout << rand() << endl;}

    //rseed(time(NULL));
    for (int i=0;i<5;i++){
        cout << randu() << endl;}

    return 0;
}
```

Hasil dua kali run, tanpa mengubah *seed* (default).

```
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$ ./rn
1804289383
846930886
1681692777
1714636915
1957747793

41
18467
6334
26500
19169
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$ ./rn
1804289383
846930886
1681692777
1714636915
1957747793

41
18467
6334
26500
19169
```

Hasil dengan mengubah seed berdasarkan waktu (`srand()` di `stdlib.h`, dan `rseed()` di `lcg.h`).

```
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$ ./rn
1458138624
1382823253
1777137677
705172191
2144641137

2315
25930
8873
2919
26891
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$ ./rn
1005141763
446532302
731286527
914087325
1432220265

2331
14136
32658
24931
12927
```

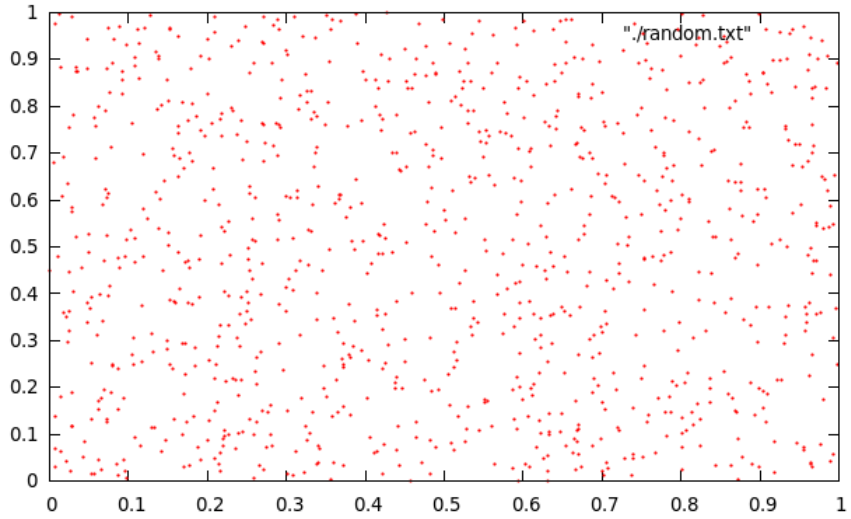
Jika kita bisa menggunakan `RAND_MAX` di `stdlib.h` (2147483647), maka pengggantinya adalah `MAX_RAND` di `lcg.h` di atas (32767).

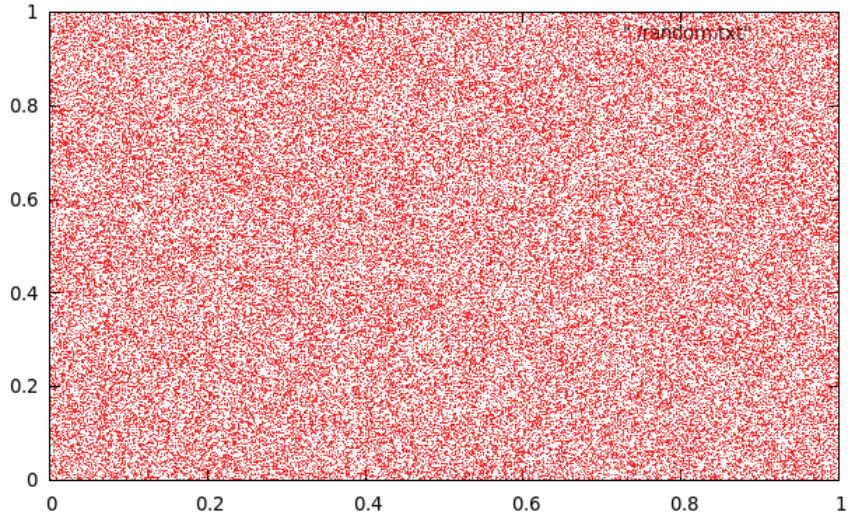
Tes *randomness* (untuk pasangan dua bilangan random).

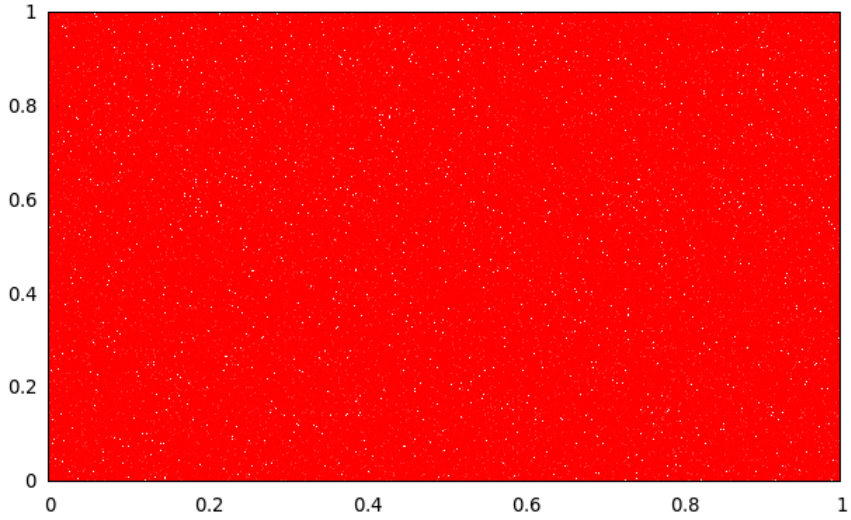
```
#include<iostream>
#include<fstream>
#include<time.h>
#include "lcg.h" // menggunakan fungsi random sendiri (microsoft)
using namespace std;

double unirand(){return (double)randu()/(double)MAX_RAND;} // rentang 0-1

int main(){
    int N = 1000;
    rseed(time(NULL));
    ofstream out("random.txt");
    for (int i=0;i<N;i++){
        out << unirand() << " " << unirand() << endl;}
    out.close();
    return 0;
}
```







1000, 10^5, dan 10^6 titik.

Integral tentu dengan metode MonteCarlo
Seperti melempar darts, algoritma yang digunakan untuk menghitung integral tentu $1D$ dapat dibuat seperti berikut ini.
Algoritma:

1. Cari maksimum/minimum dari fungsi $f(x)$ (yaitu c) pada selang a dan b (a dan b adalah batas integrasi).
2. Lakukan hingga N_{total}:
   a) random titik untuk x dengan batas a sampai b
   b) periksa apakah nilai $f(x)$ positif atau negatif
      - Jika $f(x)$ positif :
        - random untuk y dengan batas 0 sampai c.
        - periksa apakah y berada di bawah kurva $f(x)$, jika iya $N = N+1$
      - Jika $f(x)$ negatif :
        - random untuk y dengan batas 0 sampai $-c$
        - periksa apakah y berada di atas kurva $f(x)$, jika iya $N = N-1$
3. $Hasil = \frac{N}{N_{total}} \times c \times (b-a)$

dengan cara seperti di atas, kita dapat mengantisipasi apabila batas integrasi meliputi fungsi yang bernilai negatif (dapat dibuat untuk dimensi lebih tinggi pula). Pencarian nilai maksimum dapat juga dilakukan dengan cara menggunakan bilangan random (pendekatan).

Untuk fungsi $f(x) = \sqrt{1-x^4}$
- batas integrasi hanya diperbolehkan untuk $-1 \leq x \leq 1$.
- nilai maksimum di tentukan = 1 (program terlampir)

Hasil untuk $\int_{-1}^{1} \sqrt{1-x^4}dx$
Numerik = 1.748038369528061
Monte Carlo:

run_program	$N_{tot} = 10^4$	$N_{tot} = 10^5$	$N_{tot} = 10^6$
run_01	1.739600	1.748760	1.749008
run_02	1.743200	1.748880	1.748966
run_03	1.745600	1.746940	1.746786
run_04	1.745400	1.742600	1.746482
run_05	1.748200	1.749440	1.747034
average	1.7444	1.747324	1.7476552
stdev	0.003215587	0.00280287	0.0012314687

Untuk fungsi $f(x) = \sin(\sqrt{x})$
- batas integrasi hanya untuk $x > 0$.
- nilai maksimum dicari dengan bilangan random (walaupun sudah jelas = 1) (program terlampir).

Hasil untuk $\int_0^{50} \sin(\sqrt{x})dx$
Numerik = -8.557403169252579
Monte Carlo:

run_program	$N_{tot} = 10^4$	$N_{tot} = 10^5$	$N_{tot} = 10^6$
run_01	-8.068500	-8.482650	-8.668000
run_02	-8.701000	-8.485950	-8.560695
run_03	-8.827500	-8.653150	-8.569770
run_04	-8.503000	-8.394650	-8.625210
run_05	-7.920000	-8.772500	-8.602660
average	-8.404	-8.55778	-8.605267
stdev	0.395024841	0.1521640381	0.0435506865

Untuk fungsi $f(x) = e^{x^2}$
- batas integrasi hanya untuk $|x| < 26$ ($f(x)$ akan melebihi floating point).
- nilai maksimum dicari dengan f(x) pada batas terbesar (program terlampir).

Hasil untuk $\int_{-2}^{2} e^{x^2}dx$
Numerik = 32.905255531014461
Monte Carlo:

run_program	$N_{tot} = 10^4$	$N_{tot} = 10^5$	$N_{tot} = 10^6$
run_01	32.2347	32.9358	32.9316
run_02	32.5623	33.2677	32.9847
run_03	32.0819	32.5602	32.9035
run_04	33.4141	33.0821	32.9768
run_05	33.8945	32.6846	32.9275
average	32.8375	32.90608	32.94482
stdev	0.784179635	0.2878027571	0.0346237924

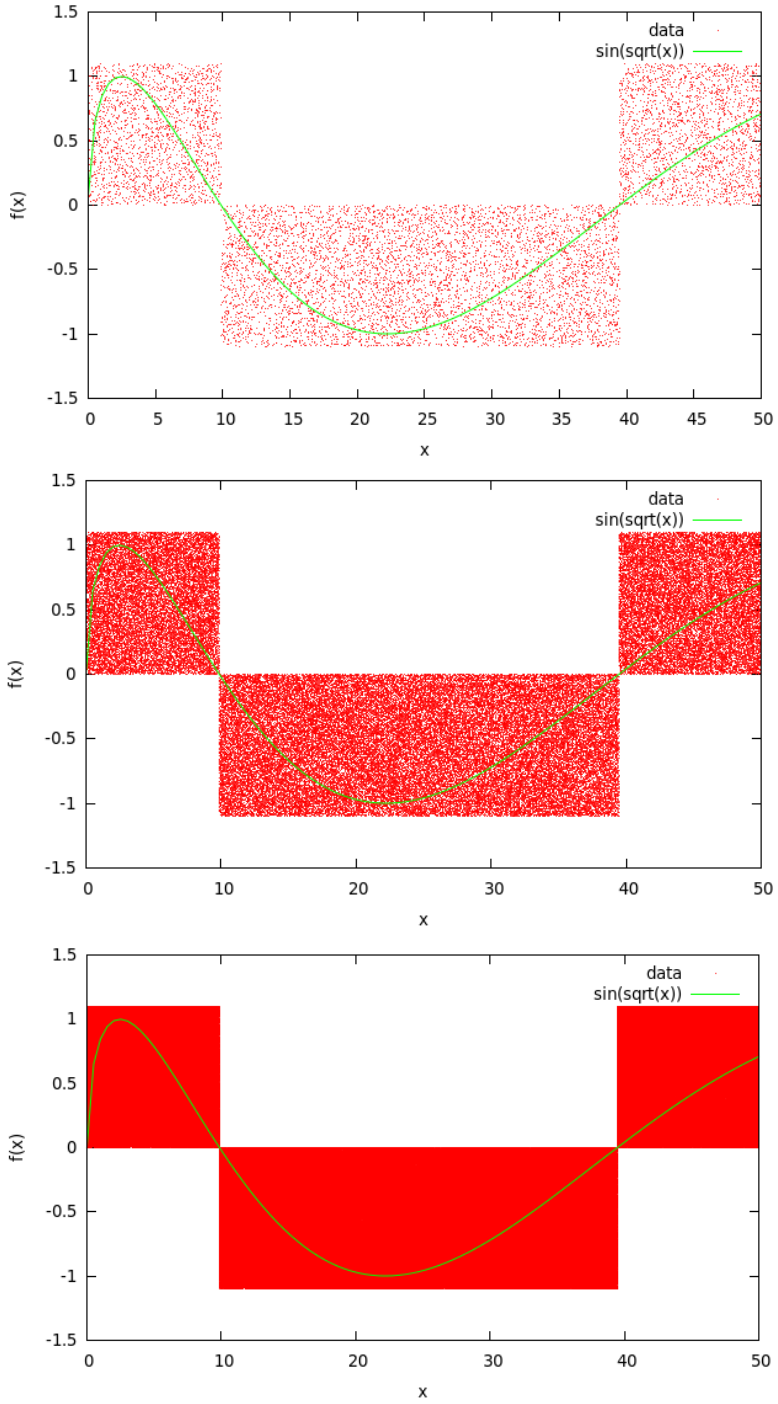
Diskusi:

1. Semakin banyak jumlah titik random dan percobaan, maka semakin presisi hasil yang diperoleh (*stdev* mengecil), walaupun tidak menjamin hasilnya lebih akurat.
2. Metode Monte Carlo akan mudah diterapkan untuk kasus integral lipat banyak (multiple integral).
3. Untuk permasalahan kompleks dapat dilakukan pembagian part integrasi dan melakukan perhitungan dengan menggunakan banyak komputer sekaligus (parallel computing).
4. Pelopor metode heuristic untuk memecahkan kasus-kasus lain.

Contoh *run* dan *error handling*:

```
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$ ./f2
=== Definite Integral for f(x) = sin(sqrt(x))
=== using MonteCarlo method
Enter the boundary (x>=0):
Lower (a) = -2
Upper (b) = 50
N_total   = 10000
Error input boundary for this function
Enter the boundary (x>=0):
Lower (a) = 50
Upper (b) = 2
N_total   = 10000
Error input boundary for this function
Enter the boundary (x>=0):
Lower (a) = 0
Upper (b) = 50
N_total   = 10000
Hasil integrasi = -8.052000
ridlo@lockon-PC:~/kul/PengenalanSK/tugasRandomGen$
```

Contoh plot (untuk $\int_0^{50} \sin(\sqrt{x})dx$) :



10^4, 10^5, dan 10^6 titik.

Lampiran

Program tentunya menggunakan fungsi random yang sudah dibuat `lcg.h`.

f1.cpp

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <fstream>
#include "lcg.h"
using namespace std;

int N, Ntot;
double x, y, res, a, b, c=1.0;

double unirand(){return (double)randu()/(double)MAX_RAND;}
double fx(double x){ return sqrt(1.-pow(x,4));}
void input();

int main(){
    cout << "=== Definite Integral for f(x) = sqrt(1-x^4)\n";
    cout << "=== using MonteCarlo method\n";
    input();
    while (a < -1. || a > 1. || b < -1. || b > 1. || b < a || a == b){
        cout << "Error input boundary for this function\n";
        input();}

    // montecarlo
    rseed(time(NULL));
    ofstream out("f1-out.txt");
    for (int i=0;i<Ntot;i++){
        x = a + (b-a)*unirand();
        y = c*unirand();
        if (y<=fx(x)){ N++;}
        out << x << " " << y << endl;}
    out.close();
    res = (double)N/(double)Ntot * c * (b-a);
    printf("Hasil integrasi = %f\n", res);

    return 0;
}

void input(){
    cout << "Enter the boundary (-1<=x<=1):\n";
    cout << "Lower (a) = "; cin >> a;
    cout << "Upper (b) = "; cin >> b;
    cout << "N_total   = "; cin >> Ntot;
}
```

f2.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <fstream>
#include "lcg.h"
using namespace std;

int N=0, Ntot=0;
double x, y, res, a, b, c=0.0;

double unirand(){return (double)randu()/(double)MAX_RAND;}
double fx(double x){ return sin(sqrt(x));}
void input();
void plot();

int main(){
    cout << "=== Definite Integral for f(x) = sin(sqrt(x))\n";
    cout << "=== using MonteCarlo method\n";
    input();
    while (a < 0. || b < 0. || b < a || a == b || Ntot < 2){
        cout << "Error input boundary for this function\n";
        input();}

    // mencari maksimum dengan bilangan random
    for (int i=0;i<Ntot;i++){
        x = a + (b-a)*unirand();
        if (fabs(fx(x)) > c) {c = fabs(fx(x));}}
    c = c + 0.1; // agar lebih menjamin c >= maksimum fungsi

    // montecarlo
    rseed(time(NULL));
    ofstream out("f2-out.txt");
    for (int i=0;i<Ntot;i++){
        x = a + (b-a)*unirand();
        if (fx(x) >= 0.){
            y = c*unirand();
            if (y <= fx(x)){ N++; }}
        else {
            y = -c*unirand();
            if (y >= fx(x)){ N--; }}
        out << x << " " << y << "\n";
    }
    out.close();

    res = ((double)N / (double)Ntot) * c * (b-a);
    printf("Hasil integrasi = %f\n", res);

    plot();
    return 0;
}
```

```
53
54 void input(){
55     cout << "Enter the boundary (x>=0):\n";
56     cout << "Lower (a) = "; cin >> a;
57     cout << "Upper (b) = "; cin >> b;
58     cout << "N_total   = "; cin >> Ntot;
59 }
60
61 void plot(){
62     ofstream ploter("plot.in");
63     ploter << "#gnuplot input file\n";
64     ploter << "set xlabel \"x\"\n";
65     ploter << "set ylabel \"f(x)\"\n";
66     ploter << "plot \"f2-out.txt\" w d title \"data\", sin(sqrt(x)) w l\n";
67     ploter.close();
68
69     system("gnuplot -persist < plot.in");
70 }
```

f3.cpp

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <math.h>
4  #include <time.h>
5  #include <fstream>
6  #include "lcg.h"
7  using namespace std;
8
9  int N, Ntot;
10 double x, y, res, a, b, c;
11
12 double unirand(){return (double)randu()/(double)MAX_RAND;}
13 double fx(double x){ return exp(x*x);}
14 void input();
15
16 int main(){
17     cout << "=== Definite Integral for f(x) = exp(x^2)\n";
18     cout << "=== using MonteCarlo method\n";
19     input();
20     while (a < -26. || a > 26. || b < -26. || b > 26. || b < a || a == b){
21         cout << "Error input boundary for this function\n";
22         input();}
23
24     // maksimum mengambil batas, karena fungsi selalu positif
25     c = fabs(fx(b));
26     if (fabs(b) < fabs(a)){ c = fabs(fx(a));}
27
28     // montecarlo
```

```
29      rseed(time(NULL));
30      ofstream out("f3-out.txt");
31      for (int i=0;i<Ntot;i++){
32          x = a + (b-a)*unirand();
33          y = c*unirand();
34          if (y<=fx(x)){ N++;}
35          out << x << " " << y << endl;}
36      out.close();
37      res = (double)N/(double)Ntot * c * (b-a);
38      printf("Hasil integrasi = %g\n", res);
39
40      return 0;
41  }
42
43  void input(){
44      cout << "Enter the boundary (-26<=x<=26):\n";
45      cout << "Lower (a) = "; cin >> a;
46      cout << "Upper (b) = "; cin >> b;
47      cout << "N_total   = "; cin >> Ntot;
48  }
```